ECDR212/214 NT Driver

User's Manual

ECHOTEK CORPORATION
555 Sparkman Drive, Suite 400
Huntsville, AL 35816
Phone: (205) 721-1911
Fax: (205) 721-9266
Web Site: http://www.echotek.com

TABLE OF CONTENTS

1) INTRODUCTION 3

2) ECDR212 DRIVER INSTALLATION 4

3) USING THE ECDR212 DRIVER DEMONSTRATION 5

BEFORE YOU START 5

4) USING THE ECDR212 DRIVER 6

GENERIC PROGRAM LAYOUT 6
PERFORMING A SYNCHRONOUS READ 6
COMPILING YOUR CODE (VISUAL C++) 6
SETTING UP AN INITIALIZATION FILE 7

5) ECDR212 DRIVER FUNCTIONS REFERENCE 9

EC_212_CLOSE 9
PURPOSE 9
SYNOPSIS 9
DESCRIPTION 9
RETURN VALUES 9
EXAMPLE 9
EC_212_IOCTL 10
PURPOSE 10
SYNOPSIS 10
DESCRIPTION 10
RETURN VALUES 11
EXAMPLE 11
EC_212_OPEN 13
PURPOSE 13
SYNOPSIS 13
DESCRIPTION 13
RETURN VALUES 13
EXAMPLE 14
EC_212_READ 15
PURPOSE 15
SYNOPSIS 15
DESCRIPTION 15
RETURN VALUES 15
EXAMPLE 16

6) ECDR212 DRIVER STRUCTURE REFERENCE 17

EC_HANDLE 17

7) ECDR212 DRIVER SUPPORT 18

A) REFERENCES 19

1) Introduction

Echotek's ECDR212/214 NT driver, henceforth referred to as the ECDR212 driver, will facilitate ease of use of Echotek's ECDR212 and/or ECDR214. This driver is supplied as a **statically linked library** that provides the following APIs:

a) *EC_212_Close* - closes the ECDR212
b) *EC_212_Ioctl* - performs various informative operations
c) *EC_212_Open* - opens the ECDR212 for reading
d) *EC_212_Read* - performs a synchronous read from the ECDR212

The source code may be provided as a courtesy; however, the tool Windrvr was used in the compilation. This means that to perform a **complete** re-compilation, Windrvr would be required. For a more detailed description of the APIs, see section 5) ECDR212 Driver Functions Reference. For hardware-level information see the hardware manual [ECH99] or [ECH00].

2) ECDR212 Driver Installation

Installation should be performed after installing the ECDR212 into the system.

a) Insert the ECDR212 NT Driver diskette into your drive.
b) Unzip “ECDR212_214_Driver.zip”
c) Copy “WINDRVR.SYS” to “c:\WINNT\SYSTEM32\DRIVERS”
d) From a DOS prompt in the directory where you installed the driver, call “WDREG.EXE install”

3) USING the ECDR212 Driver Demonstration

The code that will be used for the demonstration is a subset of the ATP for the ECDR212. A full description of the ATP is beyond the scope of this document. Rather, a single test procedure is described here. This procedure will allow you to collect data with your board without requiring an in-depth knowledge of the initialization file used or the hardware.

Before You Start

a) At least one ECDR212 must be installed in your system.
b) The ECDR212 Driver must be installed.
c) The ECDR212 in your system must be connected to a data source.
d) Your clock should be 32.5 MHz @ 10dBm.
e) Your signal should be 14.4 MHz @ 0dBm.
f) You should use 2 LP 21.4 MHz filters for optimal results.
g) Use AD1 as input.

Starting the ECDR212 Driver Demonstration

a) Explore to the “ECDR212_214 x.x/Release” directory.
b) Select “ECDR212_214_ATP.exe” and a menu will appear.
c) Select “Initialize ECDR212”.
d) Select “Program Board”.
e) You will be prompted to select an ini file. Explore to the “ECDR212_214 x.x/ini_files” directory.
f) Select “clk65fs32p5.ini”. This file performs a 14.3 MHz down convert. The setup will provide a 100kHz tone. The receiver will operate at 2x the AD Clock. The sample rate is 32.5 MHz with a decimation of 16.
g) Select “Read Functions”.
h) Start/Stop/Read Ch1 DMA.. This reads from receiver 1.
i) You will be prompted to enter the amount of data to be collected. Do not exceed the FIFO size of the board for this test. This could cause unpredictable data errors.
j) You will be prompted to save the data file. The data collected will have the MSB of each sample flipped by the demonstration code.

4) USING the ECDR212 Driver

Generic Program Layout

```
#include "EC_ECDR212.h"

void main(void)
{
        EC_HANDLE ECDR212 = NULL;

        /* Open ECDR212 receiver instance 0 */
        ECDR212 = EC_212_Open(0);
        if(ECDR212 == NULL)
                printf("ERROR: ECDR212 could not be opened\n");

        /* Perform EC_212_Ioctl and/or EC_212_ Read
           at this point in your program */

        EC_212_Close(ECDR212);
}
```

Performing a Synchronous Read

a) Call *EC_212_Ioctl* to program the ECDR212 for data collection.
b) Call *EC_212_Read* to read from the ECDR212 and note status.
c) Check that status is not *EC_ERROR* and note number of bytes read.
d) Now repeat a, b, and c as desired.

Compiling Your Code (Visual C++)

1) Include EC_ECDR212_214.lib, EC_Shared.lib, and EC_Util.lib in included libraries.
2) Include EC_ECDR212.h in all source code and in included header files.

Setting up an initialization file

You will need to create your own initialization file in order to configure the ECDR212 for your purposes. You may copy an already existing initialization file from the "ECDR212_214 x.x/ini_files" directory and modify it. The following is the set of variables in the initialization file and a list of the options available for each variable.

The following variables are entered only once		
RECEIVER_1_ENABLE	TRUE or FALSE	
RECEIVER_1_INPUT	AD_1 or AD_2	
RECEIVER_1_COLLECTION_MODE	SINGLE_6620, DUAL_6620, or QUAD_6620	
RECEIVER_1_FULL_INT_EN	TRUE or FALSE	
RECEIVER_2_ENABLE	TRUE or FALSE	
RECEIVER_2_INPUT	AD_1 or AD_2	
RECEIVER_2_COLLECTION_MODE	SINGLE_6620, DUAL_6620, or QUAD_6620	
RECEIVER_2_FULL_INT_EN	TRUE or FALSE	
RECEIVER_CLOCK_FREQ_RANGE	HIGH, MED, or LOW	
RECEIVER_CLOCK_MULT_SEL	1, 2, or 4	
COHERENT_SAMPLING	TRUE or FALSE	
ACQUISITION_MODE	SYNC or GATE	
TRIGGER_MODE	SW or HW	The trigger mode may be either software or hardware.
ACQ_CLOCK_CONFIG	RCV_CLK_GT_ACQ_CLK, or RCV_CLK_EQ_ACQ_CLK	The receiver clock may either be greater than the acquisition clock or equal to the acquisition clock.
EXT_ACQ_GATE_SEL	CLOCK_INV, CLOCK_NO_INV	The clock may either be inverted or non-inverted.
BURST_SAMPLE_COUNT	A value greater than 0	
The following variables are entered once for each channel		
DUAL_CHAN_REAL_INPUT	TRUE or FALSE	
SINGLE_CHAN_COMPLEX_INPUT	TRUE or FALSE	
SYNC_MASTER_SLAVE	MASTER or SLAVE	
NCO_BYPASS	BYPASS or ACTIVE	
PHASE_DITHER	ON or OFF	
AMPLITUDE_DITHER	ON or OFF	
RECEIVER_NCO_SYNC	This may be any hex value	The 6620 documentation recommends using FFFFFFFF under most circumstances.
NCO_FREQ	The frequency in MHz	
NCO_SAMPLE_RATE	The sample rate in MHz	
RECEIVER_NCO_PHASE_OFFSET_REG	A value greater than 0	

The following variables are entered once for each channel		
CIC2_SCALE_FACTOR	A value between 0 and 6	
EXPONENT_SHIFT	SHIFT_DOWN, or SHIFT_UP	
EXPONENT_OFFSE	A value between 0 and 7	
RECEIVER_CIC2_FILTER_DEC	A value between 1 and 16	The register takes decimation minus one; however, this is handled by the driver.
CIC5_SCALE_FACTOR	A value between 0 and 20	
RECEIVER_CIC5_FILTER_DEC	A value between 1 and 32	The register takes decimation minus one; however, this is handled by the driver.
OUTPUT_SCALE_FACTOR	A value between 0 and 7	
RECEIVER_RCF_FILTER_DEC_REG	A value between 1 and 32	The register takes decimation minus one; however, this is handled by the driver.
RCF_ADDRESS_OFFSET	A value between 0 and 255	
RECEIVER_RCF_FILTER_TAP_REG	A value between 0 and 255 for single channel real mode or a value between 0 and 128 for diversity channel real mode	The register takes taps minus one; however, this is handled by the driver.
NUM_COEFF	The number of coefficients used	
COEFF	A coefficient	This will be repeated once for every coefficient in the list.

5) ECDR212 Driver Functions Reference

EC_212_Close

PURPOSE

To release the resources associated with a particular ECDR212.

SYNOPSIS

```
#include <EC_ECDR212.h>

EC_ECDR212 EC_212_Close(EC_HANDLE hEC);
```

DESCRIPTION

EC_212_Close releases the resources associated with a particular ECDR212.

Exits immediately if hEC is NULL.

RETURN VALUES

NULL

EXAMPLE

```
void FOO(EC_HANDLE ECDR212)
{
.
.
.
      ECDR212 = EC_212_Close(ECDR212);
.
.
.
}
```

EC_212_Ioctl

PURPOSE

To perform miscellaneous access to and control of a particular ECDR212. EC_212_Ioctl takes a variable number of arguments as described below.

SYNOPSIS

```
#include <EC_ECDR212.h>

STATUS EC_212_Ioctl(EC_HANDLE hEC,
                    DWORD     Command,
                    DWORD     Arg1,
                    DWORD     Arg2,
                    ...);
```

DESCRIPTION

EC_212_Ioctl performs miscellaneous access to and control of a particular ECDR212. Primarily used to support debug efforts; however, EC_212_Ioctl must be used as described below to program the board.

The following commands are available:

EC_ECDR212_PROGRAM – programs the ECDR212; Arg1 is an initialization file name (see section 4) USING the ECDR212 Driver-Setting up an initialization file); no other arguments are used

EC_CFG_READ - reads a PCI configuration register; Arg1 is the register offset to be read; Arg2 is an address to which the register contents will be stored; no other arguments are used

EC_PROBE_PLX - reads a PLX register; Arg1 may be READ which indicates that a register will be read or WRITE which indicates that a register will be written; Arg2 is the register offset to be either read or written; Arg3 is either an address to which the register contents will be stored or the address of the value to be written to the register; no other arguments are used

EC_PROBE_ECDR212 - reads an ECDR212 register; Arg1 may be READ which indicates that a register will be read or WRITE which indicates that a register will be written; Arg2 is the register offset to be either

```
                        read or written; Arg3 is either
                        an address to which the register
                        contents will be stored or the
                        address of the value to be written
                        to the register; no other arguments
                        are used
        EC_PROBE_CH1_FIFO - reads one word from the channel
                         1 FIFO; Arg1 is an address to
                         which the FIFO contents will be
                         stored; no other arguments are
                         used
        EC_PROBE_CH2_FIFO - reads one word from the channel
                         2 FIFO; Arg1 is an address to
                         which the FIFO contents will be
                         stored; no other arguments are
                         used
```

RETURN VALUES

On success, EC_OKAY.

Otherwise, EC_ERROR is returned.

EXAMPLE

```
void FOO(EC_HANDLE ECDR212)
{
.
.
.
      DWORD data;

      /* Program the ECDR212 */
      Status = EC_212_Ioctl(ECDR212, EC_ECDR212_PROGRAM,
                            "/home/ecdr212.ini");
      if(Status == EC_ERROR)
            printf("ERROR\n");

      /* Read PCI configuration header offset 0 on the ECDR212 */
      /* storing the value to data                             */
      Status = EC_212_Ioctl(ECDR212, EC_CFG_READ, 0, &data);
      if(Status == EC_ERROR)
            printf("ERROR\n");

      /* Read PLX register offset 0 on the ECDR212 storing the */
      /* value to data                                         */
      Status = EC_212_Ioctl(ECDR212, EC_PROBE_PLX, READ, 0
                            &data);
      if(Status == EC_ERROR)
            printf("ERROR\n");

      /* Write data to PLX register offset 0 on the ECDR212    */
      Status = EC_212_Ioctl(ECDR212, EC_PROBE_PLX, WRITE, 0
                            &data);
      if(Status == EC_ERROR)
            printf("ERROR\n");
```

```
        /* Read register offset 0 on the ECDR212 storing the     */
        /* value to data                                         */
        Status = EC_212_Ioctl(ECDR212, EC_PROBE_ECDR212, READ, 0
                              &data);
        if(Status == EC_ERROR)
              printf("ERROR\n");

        /* Write data register offset 0 on the ECDR212           */
        Status = EC_212_Ioctl(ECDR212, EC_PROBE_ECDR212, WRITE, 0
                              &data);
        if(Status == EC_ERROR)
              printf("ERROR\n");

        /* Read a word from the ECDR212 channel 1 FIFO storing   */
        /* it to data                                            */
        Status = EC_212_Ioctl(ECDR212, EC_PROBE_CH1_FIFO, &data);
        if(Status == EC_ERROR)
              printf("ERROR\n");

        /* Read a word from the ECDR212 channel 2 FIFO storing   */
        /* it to data                                            */
        Status = EC_212_Ioctl(ECDR212, EC_PROBE_CH2_FIFO, &data);
        if(Status == EC_ERROR)
              printf("ERROR\n");

.
.
.
}
```

EC_212_Open

PURPOSE

To allocate the resources associated with a particular ECDR212.

SYNOPSIS

```
#include <EC_ECDR212.h>

EC_HANDLE EC_212_Open(DWORD Instance);
```

DESCRIPTION

EC_212_Open allocates the resources associated with a particular instance of an ECDR212. This Instance is simply an integer. For example:

A system with 1 ECDR212: Instance is always 0

A system with 2 ECDR212s: Instance is 0 for the ECDR212 in the first PMC site and 1 for the ECDR212 in the second PMC site.

A system with n ECDR212s: Instance is 0 for the ECDR212 in the first PMC site, 1 for the ECDR212 in the second PMC site, and n-1 for the ECDR212 in the nth PMC Site.

RETURN VALUES

On success, an EC_HANDLE to a particular instance of an ECDR212.

Otherwise, NULL is returned.

EXAMPLE

```
void FOO(EC_HANDLE ECDR212)
{
.
.
.
      /* Open ECDR212 instance */
      ECDR212 = EC_212_Open(0);
      if(ECDR212 == NULL)
            printf("ERROR:  ECDR212 could not be opened\n");
.
.
.
}
```

EC_212_Read

PURPOSE

To read data synchronously from an ECDR212.

SYNOPSIS

```
#include <EC_ECDR212.h>

DWORD EC_212_Read(EC_HANDLE hEC,
                  DWORD     *Buffer_Address,
                  DWORD     Word_Count,
                  DWORD     Channel_No);
```

DESCRIPTION

EC_212_Read reads Word_Count from Channel_No associated with hEC to Buffer_Address. The data format is dependent on the data order switch on the board. When position 1 of this switch is “ON”, the data is read from the board with the I as the lower word and the Q as the upper word. This data order is reversed if position 1 is “OFF”. See the hardware manual [ECH99] or [ECH00] for more details.

RETURN VALUES

On success, EC_OKAY is returned.

Otherwise, EC_ERROR is returned.

EXAMPLE

```
void FOO(EC_HANDLE ECDR212)
{
      DWORD Read_Buffer[BUFFER_SIZE];
      DWORD Status = EC_OKAY;
.
.
.
      /* Read from channel 0 on an ECDR212 */
      Status = EC_212_Read(ECDR212, Read_Buffer, BUFFER_SIZE, 0);
      if(Status == EC_ERROR)
            printf("ERROR\n");
      else
            printf(“Number of bytes read %d\n”, Status);
.
.
.
}
```

6) ECDR212 Driver Structure Reference

EC_HANDLE

EC_HANDLE is a driver-related pointer to a structure containing ECDR212 information. It is used to keep track of information about current ECDR212 usage. *EC_HANDLE* will be used whenever a driver function is called. *EC_212_Open* allocates and initializes *EC_HANDLE* and *EC_212_Close* frees it. EC_HANDLE is defined in the file “EC_ECDR212.h”.

7) ECDR212 Driver Support

Support can be obtained by contacting the software department via one of the following methods:

e-mail:	jane@echotek.com	
Phone:	(256)721-1911	Attn: Jane Childers
Fax:	(256)721-9266	Attn: Jane Childers
Address:	Echotek Corporation Attn: Software Department 555 Sparkman Drive, Suite 400 Huntsville, AL 35816	

A) References

[ECH99] Echotek Corporation, "ECDR212 Reference Manual", 1999.
[ECH00] Echotek Corporation, "ECDR214 Reference Manual", 2000.